Current Status of Development of New VLBI Data Analysis Software

Sergei Bolotin, John M. Gipson, David Gordon, Daniel S. MacMillan

NVI, Inc.
7257D Hanover Parkway
Greenbelt, MD 20770
NASA Goddard Space Flight Center
Greenbelt, Maryland 20771 USA

20th EVGA Meeting
Bonn, Germany, March 29-31, 2011

Overview

VLBI data analysis software

New generation VLBI data analysis software

- In 2007 the IVS Working Group on VLBI data structures (IVS WG4) has been established.
- In August of 2009 The VLBI group at the NASA GSFC started the process of new VLBI data analysis software development.
- The design and architecture overview of the new software has been presented a year ago the IVS General Meeting at Hobart (Tasmania).
- In March, 2010 first strings of the source code have been written.
- The first executable that will replace a part of current CALC/SOLVE system will be ν**Solve**, an interactive part of SOLVE System that is currently using in preliminary data analysis.
- In this presentation we will cover the current status of software development process.

Software design

The software is written on **C++ programing language**.

It uses **Qt** library for high level data abstraction and system **libc**, **libm** for low level system functions.

Currently, it consists of two parts:

- **Space Geodesy Library**, where all algorithms are implemented (90% of source code);
- an executable **νSolve** – a driver that calls the library and organizes work with an end-user (10% of source code).

The GNU Build System

The new VLBI data analysis software will be distributed in sources.

To make the software distribution portable we are using **autoconf**, **automake** and **libtool** packages. These packages are parts of **GNU Build System**. The system allows:

- create a highly portable software distribution;
- adjust the distribution to local needs;
- minimize end-user effort to compile the package, e.g. the simplest way to install the software:

```
user@host:~> ./configure && make && make install
```

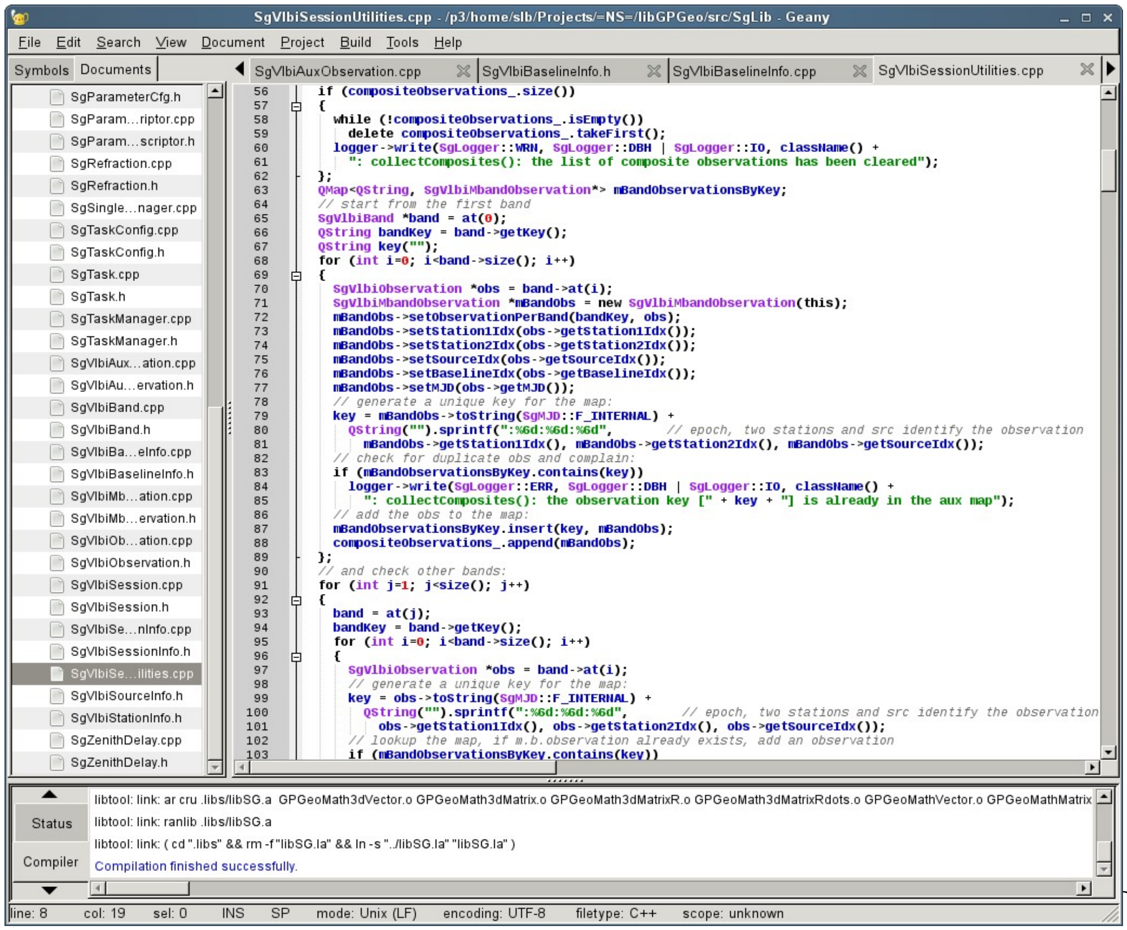
Integrated development environment

As a text editor we chose **Geany** software.

It has the following advantages:

- possess basic IDE features;
- depends on only few external packages;
- independent on distributives;
- small, lightweight and fast;
- compliant with make;

Geany is known to run under Linux, FreeBSD, NetBSD, OpenBSD, MacOS X, AIX v5.3, Solaris Express and Windows.



```
  if (compositeObservations_.size())
  {
    while (!compositeObservations_.isEmpty())
      delete compositeObservations_.takeFirst();
    logger->write(SgLogger::WRN, SgLogger::DBH | SgLogger::IO, className() +
      ": collectComposites(): the list of composite observations has been cleared");
  };
  QMap<QString, SgVlbiMbandObservation*> mBandObservationsByKey;
  // start from the first band
  SgVlbiBand *band = at(0);
  QString bandKey = band->getKey();
  QString key("");
  for (int i=0; i<band->size(); i++)
  {
    SgVlbiObservation *obs = band->at(i);
    SgVlbiMbandObservation *mBandObs = new SgVlbiMbandObservation(this);
    mBandObs->setObservationPerBand(bandKey, obs);
    mBandObs->setStation1Idx(obs->getStation1Idx());
    mBandObs->setStation2Idx(obs->getStation2Idx());
    mBandObs->setSourceIdx(obs->getSourceIdx());
    mBandObs->setBaselineIdx(obs->getBaselineIdx());
    mBandObs->setMJD(obs->getMJD());
    // generate a unique key for the map:
    key = mBandObs->toString(SgMJD::F_INTERNAL) +
      QString("").sprintf(":%6d:%6d:%6d",       // epoch, two stations and src identify the observation
        mBandObs->getStation1Idx(), mBandObs->getStation2Idx(), mBandObs->getSourceIdx());
    // check for duplicate obs and complain:
    if (mBandObservationsByKey.contains(key))
      logger->write(SgLogger::ERR, SgLogger::DBH | SgLogger::IO, className() +
        ": collectComposites(): the observation key [" + key + "] is already in the aux map");
    // add the obs to the map:
    mBandObservationsByKey.insert(key, mBandObs);
    compositeObservations_.append(mBandObs);
  };
  // and check other bands:
  for (int j=1; j<size(); j++)
  {
    band = at(j);
    bandKey = band->getKey();
    for (int i=0; i<band->size(); i++)
    {
      SgVlbiObservation *obs = band->at(i);
      // generate a unique key for the map:
      key = obs->toString(SgMJD::F_INTERNAL) +
        QString("").sprintf(":%6d:%6d:%6d",       // epoch, two stations and src identify the observation
          obs->getStation1Idx(), obs->getStation2Idx(), obs->getSourceIdx());
      // lookup the map, if m.b.observation already exists, add an observation
      if (mBandObservationsByKey.contains(key))
```

```
libtool: link: ar cru .libs/libSG.a  GPGeoMath3dVector.o GPGeoMath3dMatrix.o GPGeoMath3dMatrixR.o GPGeoMath3dMatrixRdots.o GPGeoMathVector.o GPGeoMathMatrix
libtool: link: ranlib .libs/libSG.a
libtool: link: ( cd ".libs" && rm -f "libSG.la" && ln -s "../libSG.la" "libSG.la" )
Compilation finished successfully.
```

```
line: 8    col: 19    sel: 0    INS    SP    mode: Unix (LF)    encoding: UTF-8    filetype: C++    scope: unknown
```

Geany screenshot.

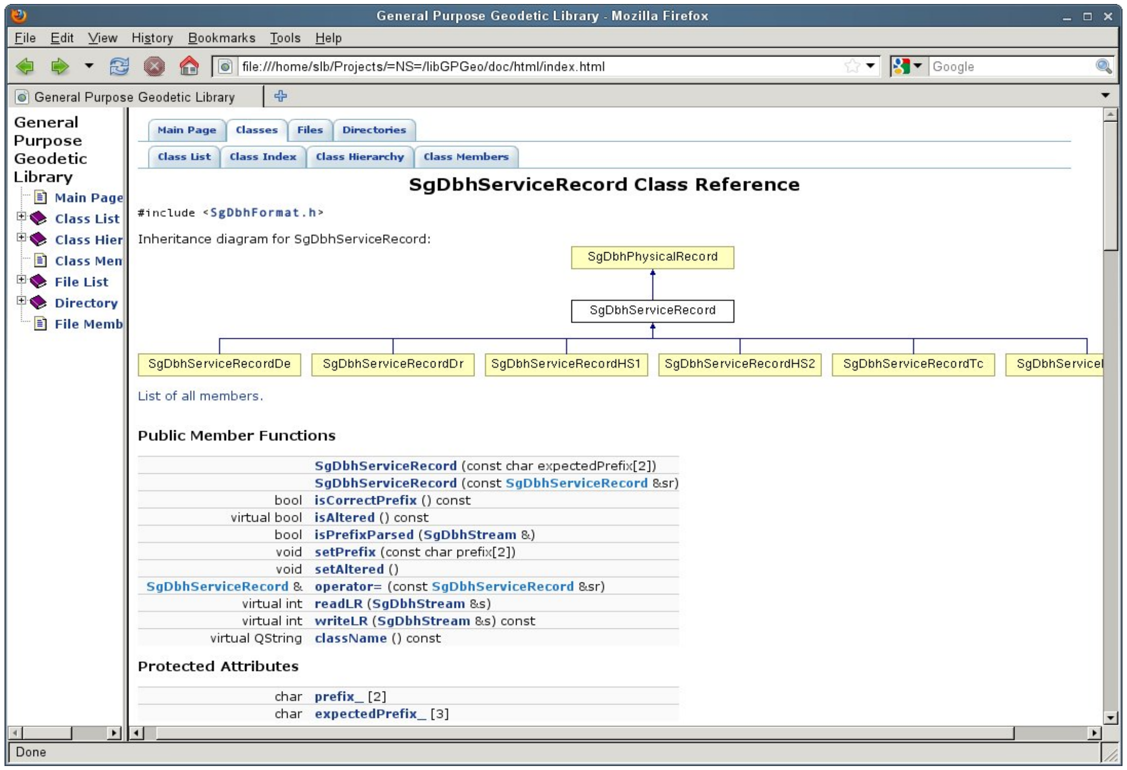
Documentation Generator

Doxygen is a documentation system for C, C++ and many other programming languages.

The software can automatically generate reference documentation in different formats:

- HTML;
- man pages;
- LaTex;
- RTF;
- PDF.

The documentation is generated directly from the sources, which makes it consistent with the current source tree.



SgDbhServiceRecord Class Reference

```
#include <SgDbhFormat.h>
```

Inheritance diagram for SgDbhServiceRecord:

List of all members.

Public Member Functions

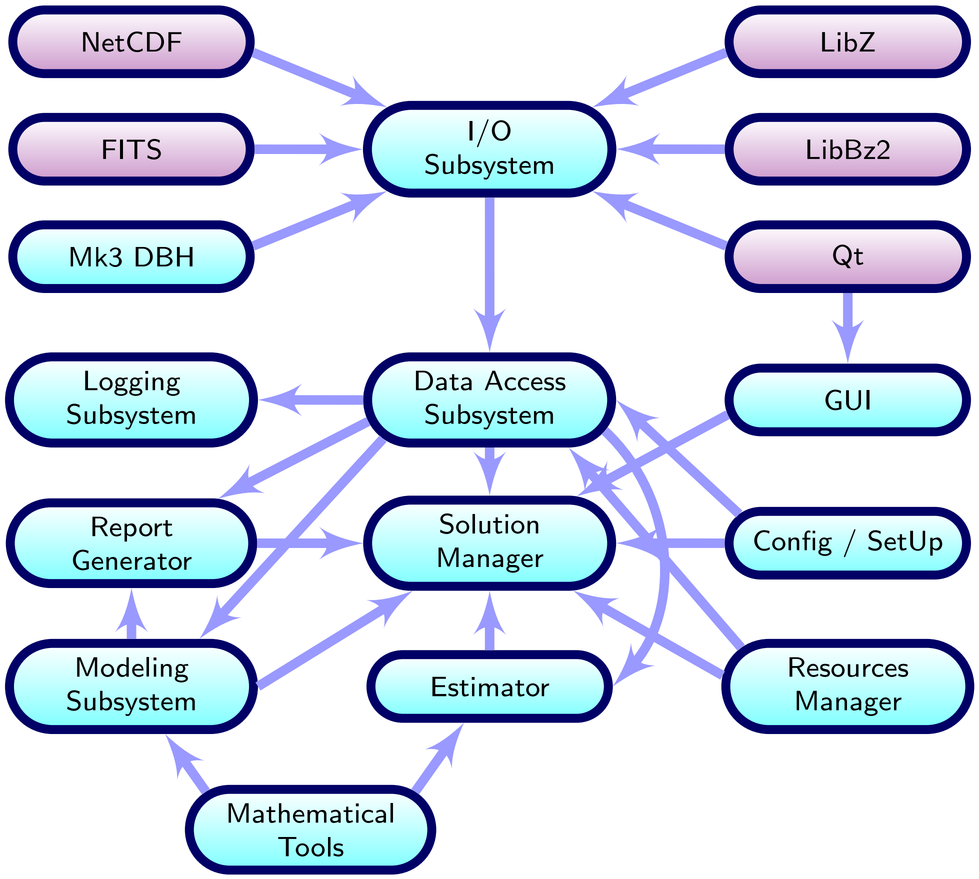
```
                      SgDbhServiceRecord (const char expectedPrefix[2])
                      SgDbhServiceRecord (const SgDbhServiceRecord &sr)
                bool  isCorrectPrefix () const
        virtual bool  isAltered () const
                bool  isPrefixParsed (SgDbhStream &)
                void  setPrefix (const char prefix[2])
                void  setAltered ()
SgDbhServiceRecord &  operator= (const SgDbhServiceRecord &sr)
         virtual int  readLR (SgDbhStream &s)
         virtual int  writeLR (SgDbhStream &s) const
     virtual QString  className () const
```

Protected Attributes

```
                char  prefix_ [2]
                char  expectedPrefix_ [3]
```

Doxygen screenshot.

Modular structure of the software



System Decomposition

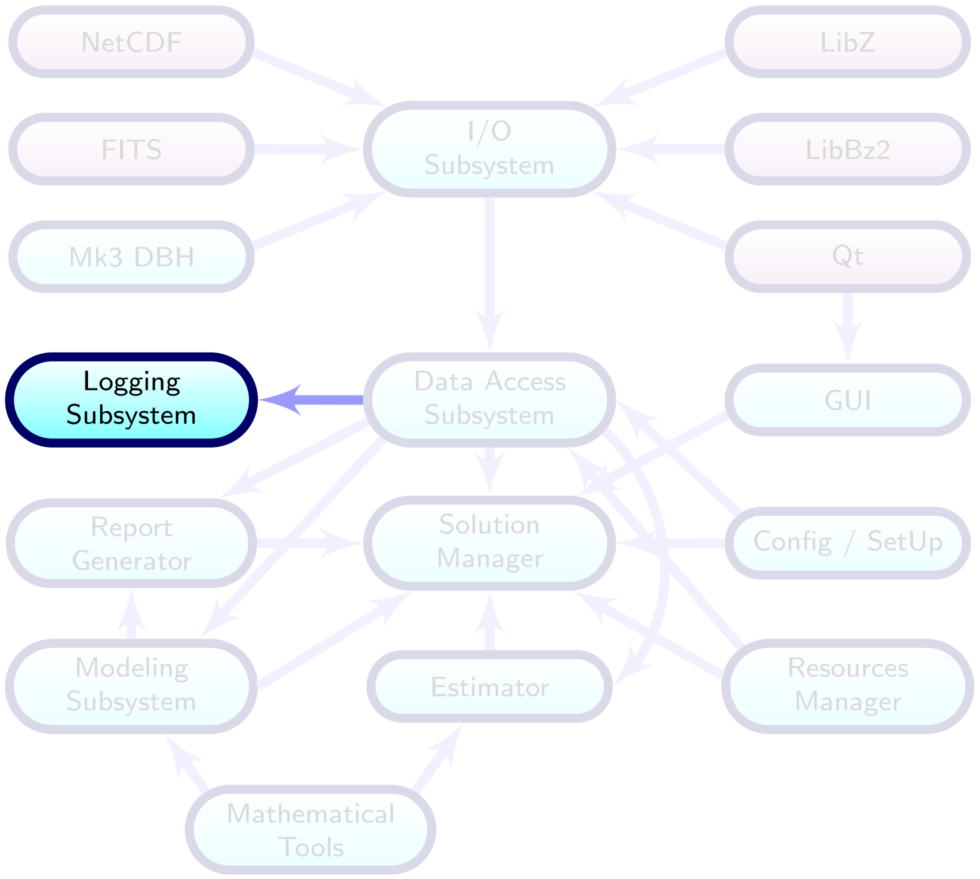
To keep our system stable and flexible

Module is a logical block of code that is loosely tied with other parts of the software.

Each arrow on the diagram represents a **dependency** or, in other words, provides information (types, function calls, constants).

Only main **dependencies** are shown on the diagram.

Modular structure of the software



Logging Subsystem

A small module that implements logging of system functionality.

There are four log levels:

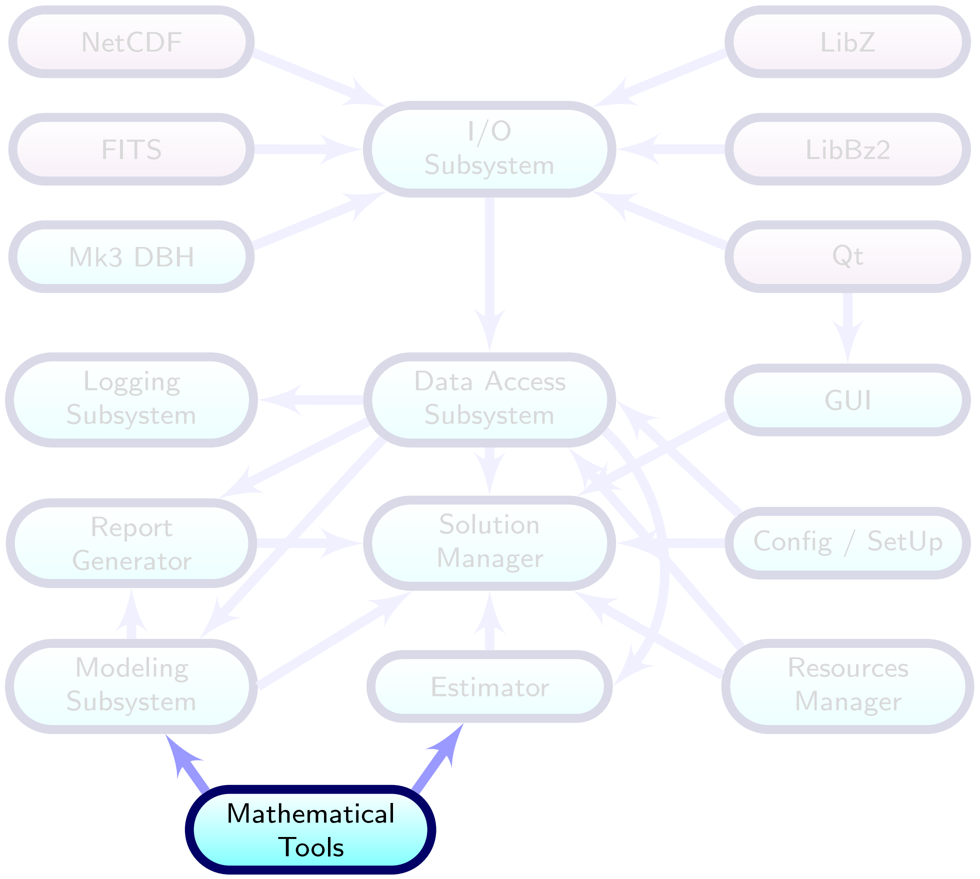
- **ERROR**
- **WARNING**
- **INFO**
- **DEBUG**

and a set of log facilities.

The **Logging Subsystem** can filter log messages by level and facility, display filtered messages and save them in a log file.

This module is used by all other modules.

Modular structure of the software



Mathematical Tools

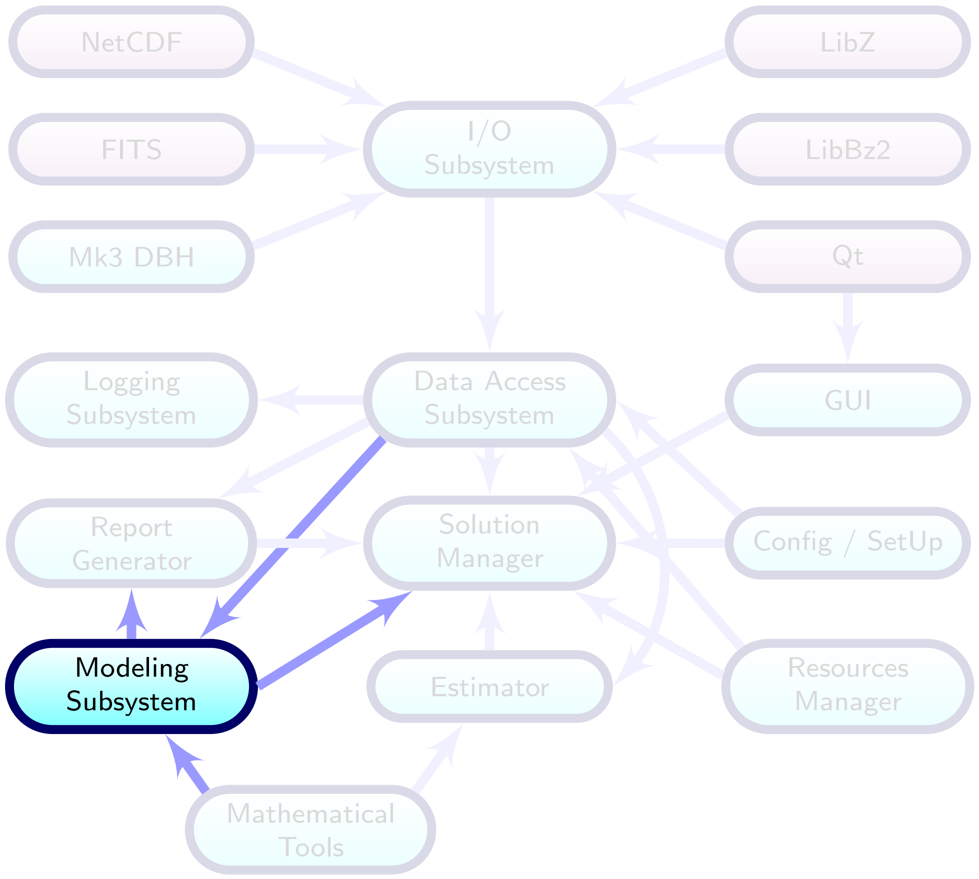
A module that contains mathematical data structures and procedures.

The following classes are implemented:

- **Vector** and **Matrix** – general classes of linear algebra;
- Specialized objects, like **Symmetric Matrix** or **Upper Triangular Matrix**, to optimize calculations;
- Geometrical classes, like **Vector3D**, **Matrix3D**, etc – to make transformations of coordinates in 3d space.

Also, optimized versions of matrix operations are realized in this module.

Modular structure of the software



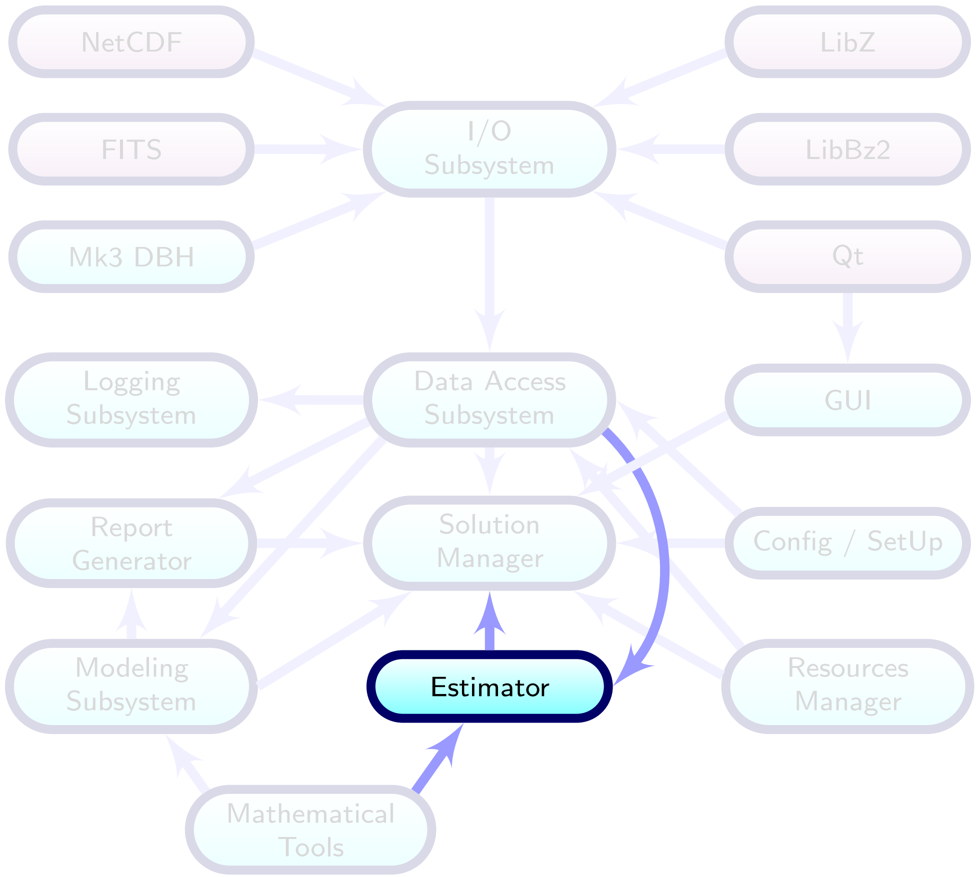
Modeling Subsystem

The main purposes of this module are: manipulation of geodetic data, computation of theoretical values and partials.

The module describes the following data structures, phenomena and models:

- **Epoch** of observation and **time interval**;
- Identities and attributes of various objects: **radio sources**, **stations** and **baselines**;
- VLBI **observation** and auxiliary data;
- Ionospheric calibration;
- Refraction correction;
- Model of clock breaks;
- Physical and geophysical constants.

Modular structure of the software



Estimator

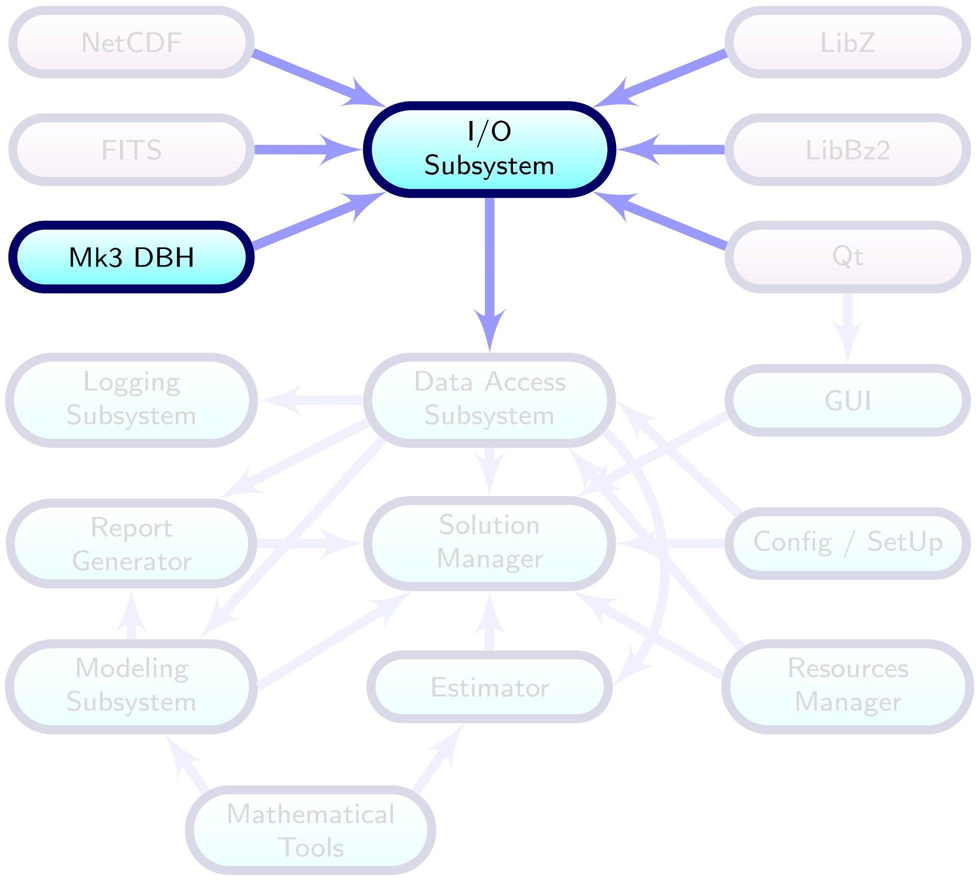
The module is responsible for performing Least Square Estimation.

Classes that are implemented in this module:

- Partial **derivative**;
- Estimated **parameter**;
- **Configurator** of parameters that will be estimated;
- **Estimator** – performs LSM estimation for lists of parameters.

Current realization implements only unbiased, session-wide parameters. Later we will add global and stochastic parameters.

Modular structure of the software



I/O Subsystem

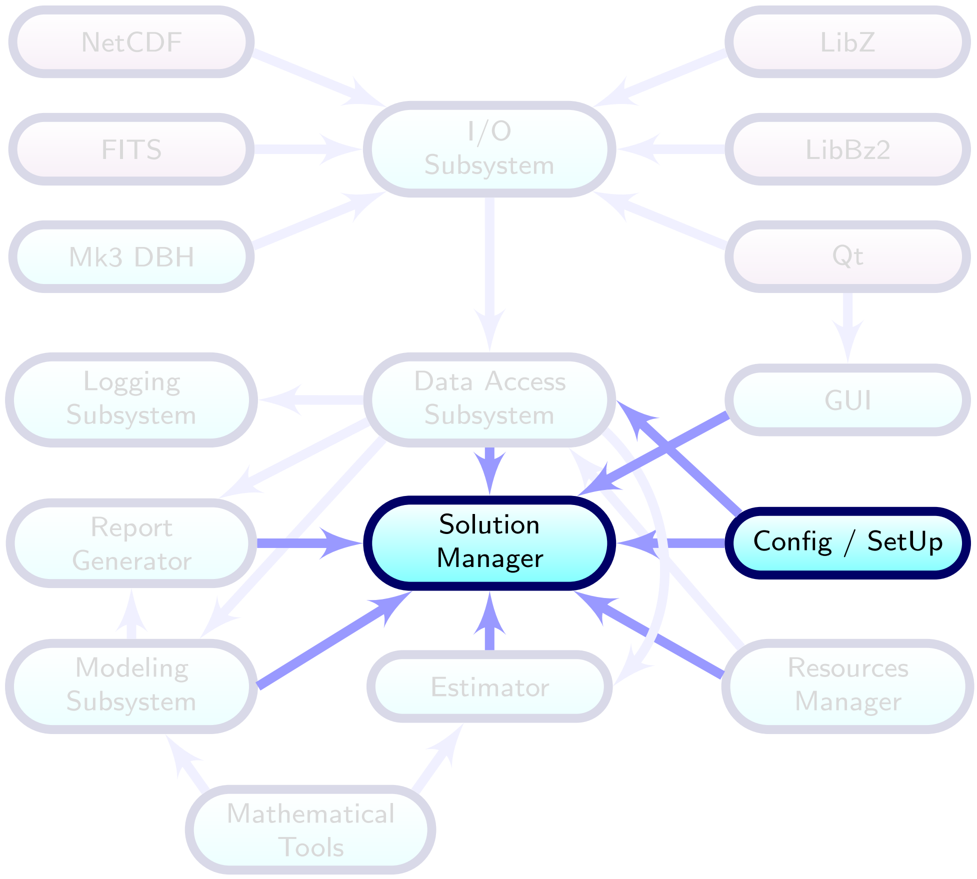
The module represents operations of input/output and supports various data formats.

Currently, only Mk3 database handling is implemented. The subsystem consists from two parts:

- A **decoder** of Mk3 DBH format;
- A logical **image** data that are stored in a Mk3 DBH file.

The module allows to read and write Mk3 DBH files and modify their structure.

Modular structure of the software



Data Flow Management

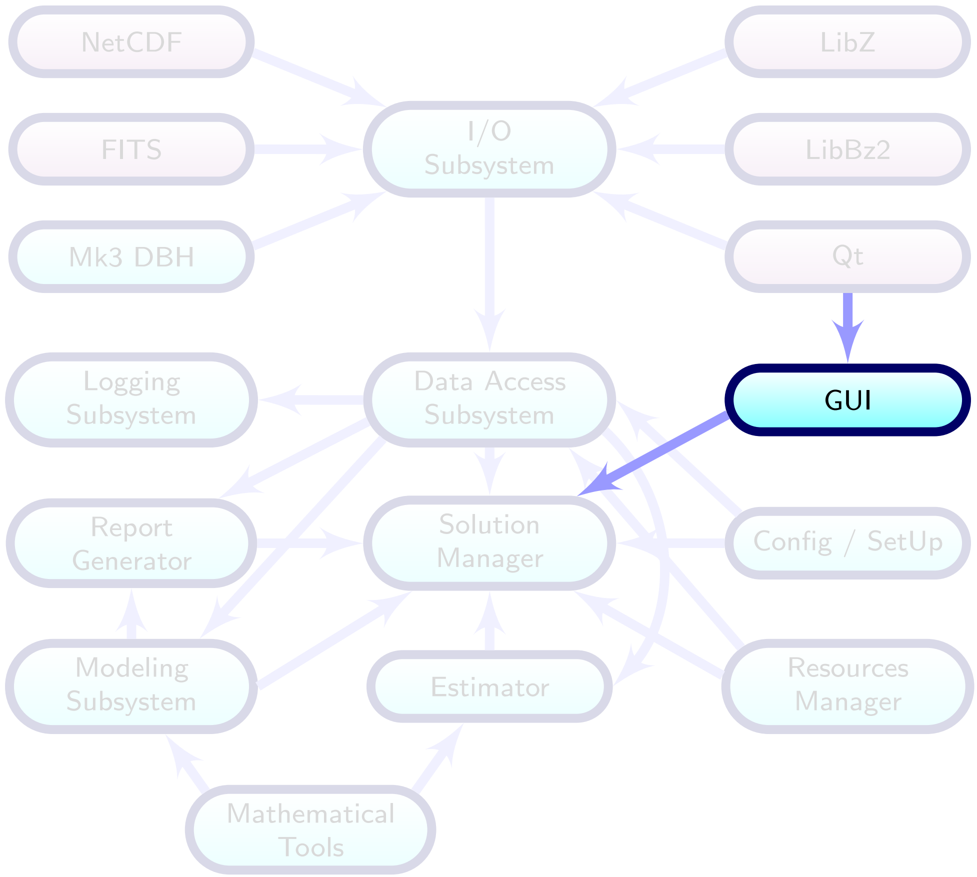
The module that performs data analysis and describes models and system configuration which should be applied in the analysis.

Classes that are implemented:

- **Configurator** – a class that describes models and algorithms;
- **Task manager** of a single session.

This module can be considered as a core of the software – it uses almost all other modules and realizes communication between different parts of the software.

Modular structure of the software



Graphical User Interface

The GUI module interacts with a user and visualize observations, solutions and auxiliary data.

Classes that are implemented in this module:

- A **Plotting Subsystem**;
- A tool for **task configuration**;
- Browsers of lists of **stations**, **baselines** and **sources**.
- etc.

This module uses widgets from **Qt** library.

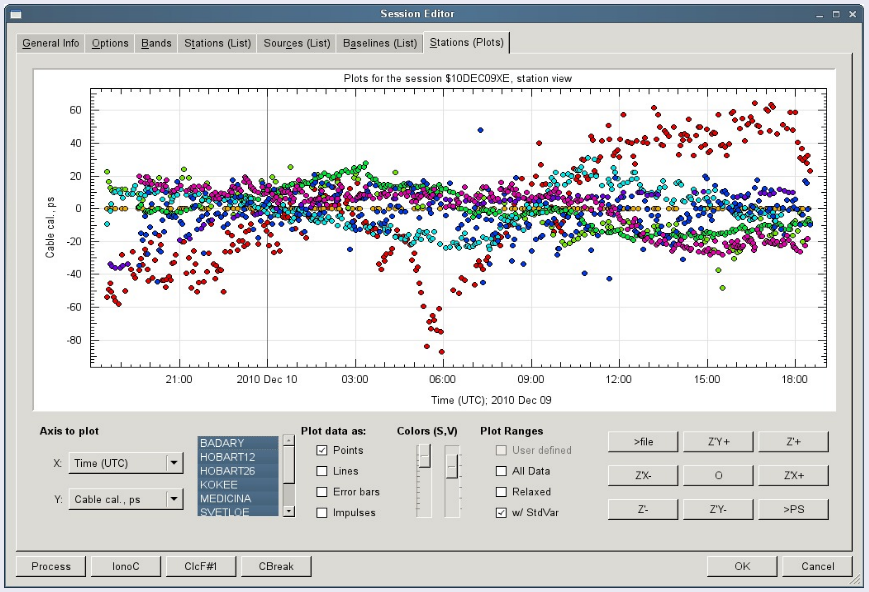
Graphical user interface

Plotting Subsystem

Features of the plotting system:

- Plots data split by branches;
- Input data can be multidimension;
- Allows user to browse "everything vs everything";
- Interacts with user, allowing user to modify data.

The plotting system is universal and can be used in various applications.



X: Time (UTC)
Y: Cable cal., ps

Plot data as: [✓ Points] Lines Error bars Impulses

Plot Ranges: User defined, All Data, Relaxed, [✓ w/ StdVar]

A screenshot of the plotting system.

General features

Current functionality

Currently, the software developing process is in the intermediate stage and not all functions are realized.

The software can:

- Read VLBI observations in Mk3 DBH format;
- Display various information that were stored in the files;
- Evaluate ionospheric corrections;
- Estimate parameters of clock functions, zenith delays, stations positions and source coordinates;
- Allow a user to resolve manually ambiguities;
- Detect and take into account clock breaks;

Ionospheric correction

Ionospheric correction

Currently, ionospheric calibration is evaluated by CALC/SOLVE system on the stage of initial processing of VLBI observations. It is introduced at the **Version 4** of Mk3 DBH files.

In new VLBI data analysis software implemented the same algorithms for evaluating effective frequencies and ionospheric corrections for **group delay**, **phase delay** and **phase delay rate**.

Clock break correction

Clock break processing

To compensate a clock break, ν**Solve** adds a step-wise linear function to the station clocks.

The following procedures are realized to process a clock break:

- Detection of a clock break: determination of the event, station and epoch;
- Evaluation of clock break value;
- Compensate clock break in the analysis.

Also, software allows to a user to add a clock break event manually.

Future

Plans

A first public release will be made in the mid of this year.

Following functions need to be implemented before the public release:

- Piecewise and stochastic estimation of clock parameters and zenith delays;
- Automatic ambiguity resolutions;
- Reweightening;
- Export observations into current CALC/SOLVE data structures.

Today during poster viewing we will have a demonstration of ν**Solve**.

Thank you for attention!